On the Convergence of Experimental Methodologies for Distributed Systems: Where do we stand?

Maximiliano Geier*†, Lucas Nussbaum† and Martin Quinson†
* Departamento de Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, Buenos Aires (C1428EGA), Argentina
† Inria, Villers-lès-Nancy, F-54600, France
Université de Lorraine, LORIA, F-54500, France
CNRS, LORIA - UMR 7503, F-54500, France

***Abstract*—Understanding distributed systems is a complex task. There are many subsystems involved, such as network equipment, disk and CPU, which effect behavior. In order to analyze this kind of applications, different approaches have been proposed: simulation, emulation and experimentation. Each paradigm has evolved independently, providing their own set of tools and methodologies.**

This paper explores how these tools and methodologies can be combined in practice. Given a simple question on a particular system, we explore how different experimental frameworks can be combined in practice. We use a representative framework for each methodology: Simgrid for simulation, Distem for emulation and Grid'5000 for experimentation. Our experiments are formally described using the workflow logic provided by the XP Flow tool.

Our long term goal is to foster a coherent methodological framework for the study of distributed systems. The contributions of this article to that end are the following: we identify a set of pitfalls in each paradigm that experimenters may encounter regarding models, platform descriptions and others. We propose a set of general guidelines to avoid these pitfalls. We show these guidelines may lead to accurate simulation results. Finally, we provide some insight to framework developers in order to improve the tools and thus facilitate this convergence.

I. Introduction

Distributed systems are pervasive in many areas of computing, ranging from scientific applications to content distribution systems. Many of these systems, such as peer-to-peer networks, comprise millions of nodes, distributed all over the world. These are generally highly heterogeneous systems, in which many different subsystems and technologies interact simultaneously using common protocols. It has been a running effort since decades to assess the properties of these systems, such as reliability, resilience, performance or security. Most often, researchers rely for that on experimentation: they analyze the behavior by running the system under a particular scenario and capturing output data that could be of interest.

Experimental work in distributed systems could be categorized in three different paradigms [1], [2]:

- **Simulation**: a prototype of the application is executed on top of a model of the environment. This approach enables the researcher to analyze questions about the system without having access to the actual environment or the actual application. The reliability of the results depend on the validity of the underlying models.
- **Emulation**: the actual application is executed on a simulated environment. The environment can be controlled through classical virtualization techniques, or a controlled artificial load can be injected onto real resources such as network links and CPUs according to the given experimental scenario.
- **Experimentation**: the actual application is executed on a real environment. Although it might be desirable, it is not always possible to do this, as it requires access to an instrumented platform that matches the real environment. This might be prohibitively expensive or not available. Moreover, testing on different scenarios under these circumstances can turn into an incredibly complex task.

Each paradigm offers its own set of tools and methodologies. Most of these tools have evolved independently from each other. It is then difficult to combine them for an augmented analysis.

This paper constitutes a status report regarding the emergence of a coherent experimental framework combining these different approaches. We opted for a practical evaluation where we conduct an experimental analysis leveraging these methodologies, and report on the difficulties encountered. To that extend, we analyze a file broadcasting application that is widely used in a cluster setting. Our focus remains however on the methodological aspect of this study, not on the results of this study per se.

This paper is organized as follows: Section II first introduces the related work while Section III describes the proposed methodology. Section IV presents several experimental traps arising in the different methodological paradigms and hints on how to avoid these traps to get satisfying results. Section V discusses several considerations that tool designers must address to facilitate the methodologies convergence. Finally, Section VI concludes this paper.

II. Related work

Several works combine differing experimental methodologies in a coherent framework toward augmented analysis. **The Emulab-Planetlab portal** [3] allows to use the interface of the Emulab [4] emulator to access the Planetlab [5] experimental platform, clearly bridging the gap between these instruments. **EMUSIM** [6] is another interesting attempt in this regard. It

integrates emulation and simulation environments in the domain of cloud computing applications, providing a framework for increased understanding of this type of systems. Similarly, **Netbed** [7] is a platform based on Emulab that mixes emulation with simulation to build homogeneous networks using heterogeneous resources.

To the best of our knowledge, there is however not much previous work that compares these methodologies in practice.

The work in [8] analyzes "myths and beliefs" about Planetlab as it stood in 2006. It concentrates on debunking assumptions about the platform that were either never correct or simply no longer true at that point. Moreover, it is clear with regard to stating real limitations of the platform, so as to help users decide if it is reasonable to use it for their objectives. This approach is different to our work in the fact that its conclusions are a set of best practices for users of a single platform. It does not analyze how Planetlab plays with other platforms and does not try to construct an unified methodology for the analysis of distributed systems.

III. Methodology

A. Selected Tools

Although certainly interesting, including in this study every existing experimental tool or framework would be a daunting task. In this work, we preferred focusing on one representative framework per methodological paradigm and focus on the methodological aspects. Additional conclusions would have been reached with other tools, but we believe that this does not reduce the impact of our conclusions.

Concerning **simulation**, we used SimGrid [9]. This is an ever growing simulation framework, with more than 10 years of development and over 100 papers based on its results. It features sound and scalable models of distributed systems. **Direct experiments** were run on Grid'5000 [10]. As of May 2013, this platform consists of 11 sites all over France and Luxembourg, with several clusters on each site, connected by high speed links. This scientific instrument is dedicated to the live study of distributed systems. To this end, it allows full deployment of custom operating system on the reserved nodes and the reservation of isolated network portions. We chose the Distem [11] **emulator**. It is easily deployed on Grid'5000 nodes, and enables the experimenter to simulate network topologies. Nodes are deployed as Linux containers, meaning that many virtual nodes could be instantiated in a single physical node, with a small resource overhead. The platform is simulated by slowing down physical links artificially (network or memory bus).

B. Driving Question

The driver of any experimental analysis is usually an interesting question that researchers are trying to answer. As a consequence, the methodology is often an afterthought, and the contribution quality comes from the results found.

This paper is rather different in that regard. As we focus on the methodology itself, we base our analysis on a simple question. It was not chosen to be innovative but instead to be simple enough to not distract the study while being complex enough to mandate non-trivial experiments. This driving question is to evaluate the relative advantages of different chain propagation algorithms for file broadcasting in a cluster setting.

One such algorithm is already implemented in the Kastafior tool of the Kadeploy suite [12]. It is used in production on Grid'5000 to deploy user OS images to each node.

It works as follows: an efficient communication chain is built to connect all participating nodes to their network neighbor when possible and to reduce the transfer interactions on the chain. This chain can be built semi-automatically since the network topology is well documented on this instrument. The file is then split in fixed size chunks, that are sent sequentially from the broadcaster to the first node of the chain. As soon as the first chunk is received by the first node, it is forwarded to the next one in the chain, concurrently to the reception of the second chunk. This algorithm is intended to minimize the time to send the whole file to all participating nodes by overlapping as much communication as possible while avoiding network interactions. In taking advantage of the network topology, this algorithm can be used to deploy files very efficiently, without using multicast or other advanced tools which introduce a huge administration overhead.

Kastafior was never analyzed thoroughly, but it performs well in practice for the users of the Grid'5000 infrastructure.

IV. Observed Traps

In this section, we show some of the problems that an experimenter might run into when analyzing a distributed system. We encountered these issues while analyzing our own implementation of the Kastafior algorithm in the context of broadcasting files in a single Grid'5000 site. This implementation, called *chainsend*, has been written entirely in C.

The metric of interest in our study is the bandwidth per node, *i.e.* the average of all bandwidths. It is measured in every node as the amount of time to *receive* the complete file divided by the file size. This value is then averaged over all nodes to get a single value. This metric has been produced by measuring time in every node. In direct experimentation in Grid'5000 and in Distem we use the local clock of each node. In the case of Simgrid, we use the simulated time provided by the framework. We show this metric as a function of the number of nodes, to indicate the progression. The data points have also been interpolated using splines to ease visualization.

Our experiments leverage up to 100 nodes of the Nancy site of Grid'5000 (clusters *griffon* and *graphene*), up to 10 virtual nodes in Distem, and 92 nodes in Simgrid (*griffon* platform file). The file size used is 1 GiB.

A. Difficulties getting the platform right

The platform in which the experiment runs tells us about network size and characteristics, how nodes are connected to each other and all the information that is required to reproduce a similar setup. However, for non-trivial experiment sizes, it could become increasingly difficult to ensure that it represents exactly what the user wanted.

In the case of Simgrid, it is important to understand the platform syntax correctly. The description is given in the form of an XML file, but it is tedious to write explicitly. In order to simplify it, some syntax shortcuts have been put in place (e.g. the `cluster` tag) which alleviate platform writing significantly, but could introduce errors if the user does not understand what they and their attributes mean exactly.

Finally, as said before, another important issue is accuracy: it is possible that the description is correct in terms of what the user wanted to say, but wrong with respect to the reference platform. This problem is significant as it could induce false conclusions from the experiments. For example, in a bandwidth-limited experiment, if the platform description is wrong with respect to this metric, it is obviously not possible to reproduce a result in simulation even though the underlying model may be correct otherwise. Latency parameters also effect metrics in unexpected ways (e.g. delaying communication and thus reducing overall bandwidth usage) and it may not be easy to identify this problem in the platform description.

Distem also shows similar problems: it is not possible to map Simgrid platform files to Distem platform descriptions yet. This is a known issue, and even though it is able to load simple *v2* Simgrid platform files correctly, most advanced features are not working yet. Moreover, there is also the non-trivial problem of mapping a virtual platform on a set of physical nodes. For example, using up several physical nodes imposes limitations on inter-node bandwidth and communication time that have to be taken into account when designing experiments. Lastly, Distem has some limitations on what kind of routing it can emulate, thus rendering impossible, for instance, to experiment with redundantly connected nodes.

B. Missing hardware models in simulation

Both simulation and emulation abstract away some details which can be found in the real platform. These abstractions could lead to wrong or inaccurate results if they are not correctly accounted for. It is important to understand them to design good experiments.

Simgrid's MSG API, for instance, forces some restrictions on what kind of software can be accurately simulated. It should be noted that, as a result of this, applications written using this API cannot be implemented as if they were actual network applications.

Simgrid simulates only a network and a CPU, the latter only given that appropriate CPU parameters were input to the model. This implies that, for example, it doesn't simulate disk activity, therefore a disk-bound application would not produce the expected results in simulation.

In Figure 1, we show a comparison of the bandwidth per node in Grid'5000 by measuring the transfer time at two different points: as soon as the file has been transferred (before the `fsync` system call is issued), and after the data has been successfully written to disk (after fsync). As we can see, *chainsend* is a disk-bound application, and as such we cannot make a fair comparison against the Simgrid implementation unless we take the disk out of the picture. Even if we measure time after fsync, there is still disk activity going on that could affect the results. In order to mitigate this, all the following runs in Grid'5000 write the file to the special device `/dev/null`. This device discards data without writing anything to disk. We show the results of doing so in Figure 2. As we can see, the results are much more stable now. The reason behind this is that there is no interference of the I/O cache, as writes are properly discarded. This scenario is more realistic compared to what is actually simulated by Simgrid.

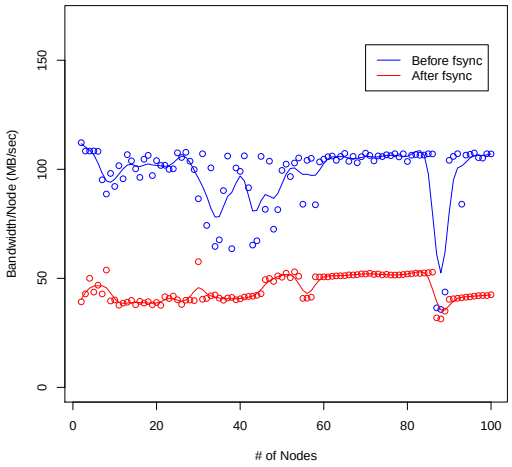

Figure 1. Bandwidth per node in Grid'5000 before fsync and after fsync.

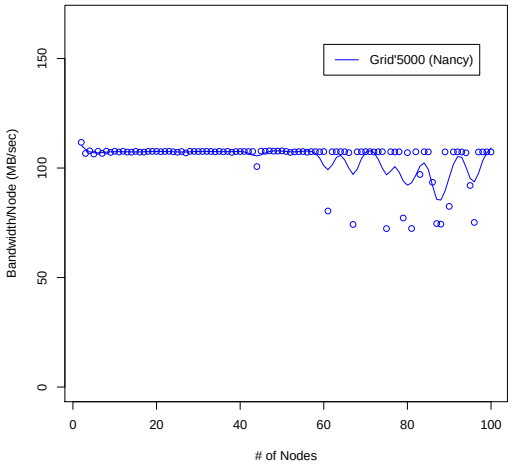

Figure 2. Bandwidth per node in Grid'5000 after discarding writes to disk.

C. Bad assumptions behind network models

Simgrid provides a network model for the simulation. This model makes some assumptions that have to be taken into account to make good use of the platform. They could be categorized as follows:

- Transport protocol: MSG assumes that all its communications are handled using TCP, ruling out every other transport protocol.
- Connection flow: the mailbox abstraction in MSG assumes that every task takes up its own connection, meaning that for each send, it simulates the time it takes to open a socket, do the three-way handshake, send data, receive its respective ACKs, and finally close it. This also implies that it can't simulate a continuous stream of data, unless it is sent as a single task.
- Connection flow arguments: there are two parameters in the default network model (LV08 [13]) that can be adjusted to change the behavior explained above, `bandwidth_factor` and `latency_factor`. The first one effects what percentage of the total bandwidth can actually be consumed by the connection, while the latter is a latency penalty factor, that effects how much time it takes to go from "slow start" to a "steady state". If this factor is closer to 1.0, the "slow start" effect is less noticeable. This factor could be used to simulate a stream of continuous data more accurately, but it is necessary to adjust it beforehand (i.e. it is not dynamically adjustable).

To highlight the effects of the connection flow parameters in Simgrid, we show bandwidth per node in Grid'5000 compared against Simgrid in Figure 3, using the default network model and connection flow arguments. As we can see, performance in Simgrid is roughly half of that in the real platform. This can be explained by what we have said before: in MSG, Simgrid simulates a complete connection for each send/receive. This means that for each file fragment being sent, there is a three-way handshake, data being sent on the network, and finally the connection is closed. At the same time, the congestion control algorithm in TCP takes place. This results in slow start taking place for every chunk of the file being sent. This is unrealistic compared to the real application, as the whole transfer happens during a single TCP connection.

If we change the default network model to CM02, which is a much simpler model that doesn't simulate accurately all the TCP congestion control algorithms, we obtain the results that are shown in Figure 4.

As we can see, the results for Simgrid are very similar to those of Grid'5000 now. This simpler model lets us fake the fact that Simgrid doesn't correctly account for single streams of data over multiple messages. In this case, it works as if there was no slow start at all, meaning that the stream is continuously in "steady state". This is very similar, although slightly overfitting, to the actual situation.

Moreover, there is an adjustable parameter, `SG_TCP_CTE_GAMMA`, which modifies how Simgrid takes into account the TCP congestion window when it updates the simulated time. This parameter can also be found in real TCP implementations (in Linux, it is possible to modify it on runtime by writing to the files `/proc/sys/net/ipv4/tcp_rmem` and `/proc/sys/net/ipv4/tcp_wmem`). These, among many other parameters, change behavior in actual TCP

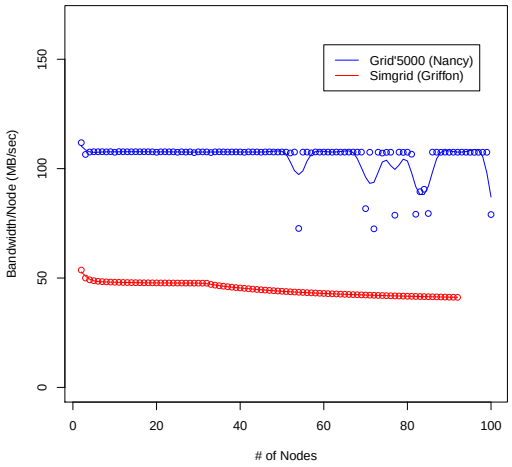

Figure 3. Bandwidth per node in Grid'5000 vs. Simgrid (default network model and arguments).

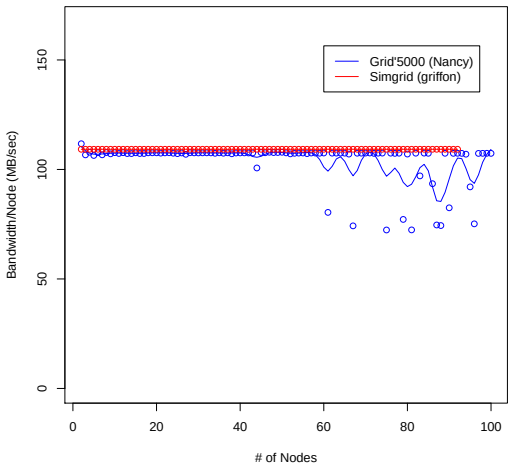

Figure 4. Bandwidth per node in Grid'5000 vs. Simgrid (CM02 network model).

implementations and it is clear that not all of them behave the same way in practice. As a consequence of this, it is very difficult to decide on which implementation to follow as the "right one", as there are so many of them on the Internet.

D. Physical node mapping interference

Distem, on the other hand, has a different set of issues arising from its abstractions. In a Distem network, the network is presented as an overlay, the underlying network and the physical nodes are hidden from the application. This derives in several constraints that have to be carefully analyzed.

One of them is the node mapping: if the overlay is emulated completely on top of a single node, memory bandwidth and system software act as a bandwidth barrier which can't be

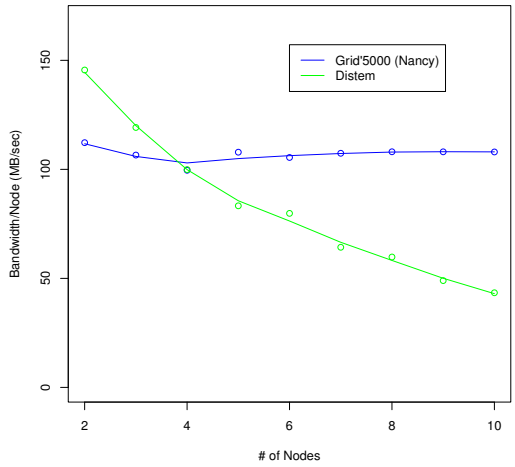

Figure 5. Bandwidth per node in Grid'5000 vs. Distem (ten virtual nodes in one physical node).

overcome. If the mapping is 1 physical node $\leftrightarrow$ 1 virtual node, the barrier exists in the network equipment that connects the nodes to each other. A simple example in this case would be that it is not possible to connect virtual nodes on a Gigabit virtual network, if the underlying physical nodes are connected at 100 Mbps. In the same sense, it is also not possible to connect 100 virtual nodes on a single physical node at 1 Gbps, if memory bandwidth at that node is only 10 Gbps, and expect consistent results. In order to visualize this problem, we can see in Figure 5 the bandwidth per node going down as the number of virtual nodes increases, when the mapping is done over one single physical node. The expected bandwidth should be much higher, but it is limited by the way kernel buffers are managed. Test runs using improved *netns* and *veth* kernel subsystems yield a higher maximum bandwidth.

Finally, bandwidth is not the only limiting factor, also emulated latencies should be higher than those of the underlying network. As an example of this, it is not possible to emulate a link with a latency of 1 μs over one with a real latency of 1 ms. Also, if both latencies are the same order of magnitude, it is very likely that there is interference from the outside network.

V. Discussion

This section discusses the traps identified in the previous section and propose some recommendations and improvements both for the experimenters and for the tool designers.

A. Platform convergence

Our first conclusion on the platform issues is that the tools should enable users to converge to the platforms used by the different approaches. In our scenario, we have three sets of platforms: Simgrid platform files, Distem platform descriptions and real network testbeds. In order to compare results among all the tools, it is necessary that the platforms represent the same scenarios, otherwise the comparison would be unfair.

The tools could help the user achieve this by checking whether they match or not. For example, in the Emulab [4] testbed, which offers a subset of the features found in Distem as used on top of Grid'5000 (namely network topology emulation by means of slowing down fast links), there is a tool called *linktest* [14], that measures link latency and bandwidth after the platform has been instantiated, in order to identify differences with respect to the platform description. This tool is useful not only to corroborate that the experimental framework is able to reproduce the input scenario, but also to ensure that the user didn't make a mistake when they designed the platform model. For example, one such link measurement tool could be run in the real platform, in the platform as emulated by Distem, and finally a simulation of the same tool on top of Simgrid, and then all the results could be compared for statistically significant differences.

Similarly, being able to use the same platform descriptions in both Simgrid and Distem would be very useful to avoid error-prone work duplication. There is still the problem of handling virtual node mappings in Distem, which is a feature that doesn't make sense in the context of Simgrid, but being able to convert from Simgrid platforms to Distem, while keeping the ability to load Simgrid platforms would be a good start. As said before, there is already work in this direction, but it still needs some refinement.

B. Identify application traits

To get the most out of the tools, it is also necessary to understand what kind of application is being analyzed. Although it might seem like a chicken-and-egg problem to have to analyze a system in order to build a better analysis for it, the user has to understand very clearly what kind of application they are working on, as to properly identify traits that would not be conveyed by the experimental framework. This means, among other things, understanding what kind of traffic the application generates, what transport protocol is used in the real implementation, what is the application protocol like, what kind of network is targeted by the application, what kind of resources could act as bottlenecks.

Most of these questions require a good understanding of the application that is being analyzed and in some cases it is enough to analyze the source code to answer them. In case it is not that easy to infer, building experiments with the framework limitations in mind is a good start.

C. Converge experimental evaluation

Tools for experiment management are great improvements in terms of being able to automate experiment setup, execution and data gathering and analysis. We have used XP Flow to build all of our experiments. This tool provides an interface to work with Grid'5000 and handle all the steps of the experiment directly. The experiments themselves are structured using workflow logic, such as that used in business to describe processes.

Building experiments by hand has many problems: it is error-prone, it doesn't scale, it becomes increasingly difficult to manage all the data as the experiment size and number grow. Working with several experimental tools expands this problem manyfold, as there is a new dimension that has to be taken into account to match similar experiments using different tools. One big problem in dealing with different tools is that they all have their own way to do things. This is worsened by the fact that there are also many tools for experiment automation. Using several tools might require writing boilerplate code and wrappers to match the way each tool works.

In order to converge to a single framework to build experiments, this code should be clearly modularized to abstract away the differences in a way that they can be written generically.

VI. Conclusions

We have evaluated three different platforms for distributed system analysis, one for each of the paradigms in experimental evaluation, concentrating on the methodological aspects. Our driving question has been the performance of chain propagation algorithms in a cluster setting. We have discussed traps that users of these platforms might run into and provided some insight on how to avoid them. In particular, we have identified problems in the accuracy of platform descriptions, missing hardware models, incorrect assumptions in network models provided by these frameworks and communication interference due to assumptions with regard to node mapping in overlays. By pointing out these problems, we have been able to create experiments to correctly assess the performance of our *chainsend* application in simulation. There is ongoing work to show the full picture, including also results in emulation. Our assessment in this case is that, due to the nature of this particular workload, emulation is not able to provide an accurate view while getting the full benefits of the platform.

Finally, an interesting topic to carry our work forward is platform validation. We plan to provide a platform validation tool similar to *linktest* for each paradigm, and also work towards the convergence of platform descriptions, in order to make it easier for the experimenter to use different tools.

It is our belief that each of the methodologies provide an unique point of view that has to be complemented in order to acquire a better understanding of the system. It is important to understand the tools and their limitations in order to use them appropriately. Our suggestions move towards making it more straightforward to manage experiments and compare results, trying to reduce repetitive, error-prone steps as much as possible.

Acknowledgments

M. Geier has a CONICET PhD fellowship. This work was supported by Inria.

References

[1] J. Gustedt, E. Jeannot, and M. Quinson, "Experimental Methodologies for Large-Scale Systems: a Survey," *Parallel Processing Letters*, vol. 19, no. 3, pp. 399–418, 2009. [Online]. Available: http://hal.inria.fr/inria-00364180

[2] O. Beaumont, L. Bobelin, H. Casanova, P.-N. Clauss, B. Donassolo, L. Eyraud-Dubois, S. Genaud, S. Hunold, A. Legrand, M. Quinson, C. Rosa, L. Schnorr, M. Stillwell, F. Suter, C. Thiery, P. Velho, J.-M. Vincent, and J. Won, Young, "Towards Scalable, Accurate, and Usable Simulations of Distributed Applications and Systems," INRIA, Rapport de recherche RR-7761, Oct. 2011. [Online]. Available: http://hal.inria.fr/inria-00631141

[3] K. Webb, M. Hibler, R. Ricci, A. Clements, and J. Lepreau, "Implementing the emulab-planetlab portal: Experience and lessons learned," in *In Workshop on Real, Large Distributed Systems (WORLDS)*, 2004.

[4] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar, "An integrated experimental environment for distributed systems and networks," Boston, MA, Dec. 2002, pp. 255–270.

[5] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "Planetlab: an overlay testbed for broad-coverage services," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 3, pp. 3–12, Jul. 2003. [Online]. Available: http://doi.acm.org/10.1145/956993.956995

[6] R. N. Calheiros, M. A. Netto, C. A. De Rose, and R. Buyya, "Emusim: an integrated emulation and simulation environment for modeling, evaluation, and validation of performance of cloud computing applications," *Software: Practice and Experience*, vol. 43, no. 5, pp. 595–612, 2013. [Online]. Available: http://dx.doi.org/10.1002/spe.2124

[7] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar, "An integrated experimental environment for distributed systems and networks," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 255–270, Dec. 2002. [Online]. Available: http://doi.acm.org/10.1145/844128.844152

[8] N. Spring, L. Peterson, A. Bavier, and V. Pai, "Using planetlab for network research: myths, realities, and best practices," in *IN PROCEEDINGS OF THE SECOND USENIX WORKSHOP ON REAL, LARGE DISTRIBUTED SYSTEMS (WORLDS)*, 2006, pp. 17–24.

[9] H. Casanova, A. Legrand, and M. Quinson, "Simgrid: a generic framework for large-scale distributed experiments," in *Proceedings of the Tenth International Conference on Computer Modeling and Simulation*, ser. UKSIM '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 126–131. [Online]. Available: http://dx.doi.org/10.1109/UKSIM.2008.28

[10] F. Cappello, E. Caron, M. Dayde, F. Desprez, Y. Jegou, P. Primet, E. Jeannot, S. Lanteri, J. Leduc, N. Melab, G. Mornet, R. Namyst, B. Quetier, and O. Richard, "Grid'5000: A large scale and highly reconfigurable grid experimental testbed," in *Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*, ser. GRID '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 99–106. [Online]. Available: http://dx.doi.org/10.1109/GRID.2005.1542730

[11] L. Sarzyniec, T. Buchert, E. Jeanvoine, and L. Nussbaum, "Design and Evaluation of a Virtual Experimental Environment for Distributed Systems," in *PDP2013 - 21st Euromicro International Conference on Parallel, Distributed and Network-Based Processing*, Belfast, Royaume-Uni, Aug. 2012, rR-8046 RR-8046. [Online]. Available: http://hal.inria.fr/hal-00724308

[12] E. Jeanvoine, L. Sarzyniec, and L. Nussbaum, "Kadeploy3: Efficient and Scalable Operating System Provisioning," *USENIX ;login:*, vol. 38, no. 1, pp. 38–44, Feb. 2013.

[13] P. Velho and A. Legrand, "Accuracy study and improvement of network simulation in the simgrid framework," in *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, ser. Simutools '09. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009, pp. 13:1–13:10. [Online]. Available: http://dx.doi.org/10.4108/ICST.SIMUTOOLS2009.5592

[14] D. Anderson, M. Hibler, L. Stoller, T. Stack, and J. Lepreau, "Automatic online validation of network configuration in the emulab network testbed," in *Autonomic Computing, 2006. ICAC '06. IEEE International Conference on*, 2006, pp. 134–142.